数学小工具

快速乘(一般不会出问题)

```
ll ksc(ll a,ll b,ll mod){
        return (a*b-(ll)((long double)a/mod*b)*mod+mod)%mod;
        //return (a*b-(ll)((long double)a/mod*b+1e-8)*mod+mod)%mod;
}
```

龟速乘

```
ll mul(ll a,ll k,ll mod){
        ll ans=0;
        while(k){
                if(k&1){
                        ans=(ans+a)%mod;
                }
                a=(a+a)%mod;
                k>>=1;
        }
        return ans;
}
```

拓展欧几里得(exgcd)

```
ll exgcd(ll a,ll b,ll &x,ll &y){
        if(b==0){
                x=1;y=0;
                return a;
        }
        ll d=exgcd(b,a%b,y,x);
        y-=a/b*x;
        return d;
}
```

中国剩余定理（excrt）

```
ll excrt(int n, ll r[], ll m[]){
        ll x = r[1], y1, y2, a = m[1];
        for (int i = 2;i <= n;i++){
                ll b = m[i], c = ((r[i] - x) % b + b) % b;
                ll d = exgcd(a, b, y1, y2);
                if (c % d)return -1;//无解
                //y1 = (__int128)y1 * (c / d) % (b / d);
                y1=ksc(y1,c/d,b/d);
                x += a * y1;
                a *= b / d;
                x = (x % a + a) % a;
        }
        return x;
}
```

矩阵快速幂

构造

1. 给定矩阵A，求$A + A^2 + A^3 + ... + A^k$的结果

   A + A^2 + A^3 + A^4 + A^5 + A^6 =(A + A^2 + A^3) + A^3*(A + A^2 + A^3)
   应用这个式子后，规模k减小了一半。我们二分求出A^3后再递归地计算A + A^2 + A^3，即可得到原问题的答案。

2. Now we define another kind of Fibonacci : $A(0) = 1, A(1) = 1, A(N) = X * A(N - 1) + Y * A(N - 2)(N >= 2)$.And we want to Calculate $S(N), S(N) = A(0)^2 + A(1)^2 + ... + A(n)^2$.

   考虑1*4 的矩阵$[s[n-2], a[n-1]^2, a[n-2]^2, a[n-1] * a[n-2]]$
   我们需要找到一个4×4的矩阵A，使得它乘以A得到1×4的矩阵$[s[n-1], a[n]^2, a[n-1]^2, a[n] * a[n-1]]$
   可以构造矩阵A为：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & x^2 & 1 & x \\ 0 & y^2 & 0 & 0 \\ 0 & 2xy & 0 & y \end{bmatrix}$$

3. 给定n个点，m个操作，构造O(m+n)的算法输出m个操作后各点的位置。操作有平移、缩放、翻转和旋转

   这里的操作是对所有点同时进行的。其中翻转是以坐标轴为对称轴进行翻转（两种情况），旋转则以原点为中心。如果对每个点分别进行模拟，那么m个操作总共耗时O(mn)。利用矩阵乘法可以在O(m)的时间里把所有操作合并为一个矩阵，然后每个点与该矩阵相乘即可直接得出最终该点的位置，总共耗时O(m+n)。假设初始时某个点的坐标为x和y，下面5个矩阵可以分别对其进行平移、旋转、翻转和旋转操作。预先把所有m个操作所对应的矩阵全部乘起来，再乘以(x,y,1)，即可一步得出最终点的位置。

$$\begin{pmatrix}1&0&p\\0&1&q\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}x\\y\\1\end{pmatrix}=\begin{pmatrix}x+p\\y+q\\1\end{pmatrix}$$

平移

$$\begin{pmatrix}L&0&0\\0&L&0\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}x\\y\\1\end{pmatrix}=\begin{pmatrix}x*L\\y*L\\1\end{pmatrix}$$

缩放

$$\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}x\\y\\1\end{pmatrix}=\begin{pmatrix}x\\-y\\1\end{pmatrix}$$

上下翻转

$$\begin{pmatrix}-1&0&0\\0&1&0\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}x\\y\\1\end{pmatrix}=\begin{pmatrix}-x\\y\\1\end{pmatrix}$$

左右翻转

$$\begin{pmatrix}\cos\alpha&-\sin\alpha&0\\\sin\alpha&\cos\alpha&0\\0&0&1\end{pmatrix}\cdot\begin{pmatrix}x\\y\\1\end{pmatrix}=\begin{pmatrix}\cos\alpha\,x-\sin\alpha\,y\\\sin\alpha\,x+\cos\alpha\,y\\1\end{pmatrix}$$

绕原点旋转

4. 给定一个有向图，问从A点恰好走k步（允许重复经过边）到达B点的方案数mod p的值

   把给定的图转为邻接矩阵，即$A(i,j)=1$当且仅当存在一条边i->j。令$C=A*A$，那么$C(i,j)=\Sigma A(i,k)*A(k,j)$，实际上就等于从点i到点j恰好经过2条边的路径数（枚举k为中转点）。类似地，$C*A$的第i行第j列就表示从i到j经过3条边的路径数。同理，如果要求经过k步的路径数，我们只需要二分求出A^k即可。

代码

```
struct matrix{
        ll a[N][N];
        int n,m;
        matrix(int _n=N-1,int _m=N-1){
                n=_n;m=_m;
                for(int i=0;i<=n;i++){
                        for(int j=0;j<=m;j++){
                                a[i][j]=0;
                        }
                }
        }
        matrix operator *(const matrix b)const{
                matrix c(n,b.m);
                for(int i=1;i<=n;i++){
                        for(int k=1;k<=m;k++){
                                if(a[i][k]==0)continue;
                                for(int j=1;j<=b.m;j++){
                                        c.a[i][j]=(c.a[i][j]+a[i][k]*b.a[k][j])%mod;
                                }
                        }
                }
                return c;
        }
        matrix operator ^(ll b)const{
                matrix c(n,n),d(n,n);
                for(int i=1;i<=n;i++){
                        c.a[i][i]=1;
                }
                for(int i=1;i<=n;i++){
                        for(int k=1;k<=n;k++){
                                d.a[i][k]=a[i][k];
                        }
                }
                while(b){
                        if(b&1)c=c*d;
                        b>>=1;
                        d=d*d;
                }
                return c;
        }
}mat[3];
```

求1-n与P互质的数字数量

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll n,p;
ll cnt_pri(ll n,ll x){
        vector<ll>p;
        for(ll i=2;i*i<=x;i++){
                if(x%i==0){
                        p.push_back(i);
                        while(x%i==0){
                                x/=i;
                        }
                }
        }
        if(x>1){
                p.push_back(x);
        }
        int len=p.size();
        ll ans=0;
        for(int i=1;i<1<<len;i++){
                ll tmp=1,t=0;
                for(int j=0;j<len;j++){
                        if(i>>j&1){
                                tmp*=p[j];
                                t++;
                        }
                }
                if(t&1){
                        ans+=n/tmp;
                }else{
                        ans-=n/tmp;
                }
        }
        return n-ans;

}



int main(){
        scanf("%lld %lld",&n,&p);
        cout<<cnt_pri(n,p);
        return 0;
}
```

线性基

异或线性基

```
struct LineBase{
        int r;
        ll a[N];
        LineBase(int _r=62){
                r=_r;
                for(int i=0;i<=r;i++)a[i]=0;
        }
        void insert(ll x){
                for(int i=r;i>=0&&x;i--){
                        if(x&(1ll<<i)){
                                if(a[i]){
                                        x^=a[i];
                                }else{
                                        a[i]=x;
                                        break;
                                }
                        }
                }
        }
        void mergeFrom(const LineBase b){
                for(int i=0;i<=r;i++)insert(b.a[i]);
        }
        ll query(ll x){
                for(int i=r;i>=0;i--){
                        if((x^a[i])>x)x^=a[i];
                }
                return x;
        }
}lb;
```

NTT

大数乘法 **a*b=c**

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=1e7+9;
const int p=998244353,g=3,gi=332748118;
typedef long long ll;
int r[maxn<<2],bit,num=1;
ll a[maxn],b[maxn];
int n,m,inv;
void butt(int num,int bit){
        for(int i=0;i<num;i++){
                r[i]=(r[i>>1]>>1)|((i&1)<<(bit-1));
        }
}
ll ksm(ll x,int n){
        ll ans=1;
        while(n){
                if(n&1)ans=ans*x%p;
                x=x*x%p;
                n>>=1;
        }
        return ans;
}
void ntt(ll a[],int opt){
        for(int i=0;i<num;i++){
                if(i<r[i])swap(a[i],a[r[i]]);
        }
        for(int mid=1;mid<num;mid<<=1){
                ll wn=ksm(opt==1?g:gi,(p-1)/(mid<<1));
                for(int R=mid<<1,j=0;j<num;j+=R){
                        ll w=1;
                        for(int k=0;k<mid;k++,w=w*wn%p){
                                ll x=a[j|k],y=w*a[j|k|mid]%p;
                                a[j|k]=(x+y)%p;
                                a[j|k|mid]=(x-y+p)%p;
                        }
                }
        }
        if(opt==-1)for(int i=0;i<num;i++)a[i]=a[i]*inv%p;
}
string str;
int main(){
        cin>>str;
        int len=str.size();
        n=len;
        for(int i=0;i<len;i++){
                a[n-i-1]=str[i]^48;
        }
        cin>>str;
        len=str.size();
```

```
        m=len;
        for(int i=0;i<len;i++){
                b[m-i-1]=str[i]^48;
        }
        for(;num<n+m;bit++,num<<=1);
        butt(num,bit);
        inv=ksm(num,p-2);
        ntt(a,1);
        ntt(b,1);
        for(int i=0;i<num;i++){
                a[i]=a[i]*b[i]%p;
        }
        ntt(a,-1);
        n=n+m;
        for(int i=0;i<n;i++){
                a[i+1]+=a[i]/10;
                a[i]%=10;
        }
        while(a[n]==0)n--;
        for(int i=n;i>=0;--i){
                printf("%lld",a[i]);
        }
        return 0;
}
```

莫队

带修莫队

一般块长$n^{2/3}$, $O(n^{5/3})$

块长$\frac{n^{2/3}*t^{1/3}}{m^{1/3}}$, $O(n^{2/3}*m^{2/3}*t^{1/3})$

** 计算区间[l,r],数字出现的次数的mex（答案最多$n^{1/2}$） **

```
//#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2e5+9;
const int mod=1e9+7;
ll read(){
        ll ans=0,f=1;
        char c=getchar();
        while(c<'0'||c>'9'){
                if(c=='-')f=-1;
                c=getchar();
        }
        while(c>='0'&&c<='9'){
                ans=(ans<<1)+(ans<<3)+c-'0';
                c=getchar();
        }
        return ans*f;
}
int m;
int bk[N];
struct Q{
        int l,r;
        int tim;
        int id;
        bool operator<(const Q x)const{
                if(bk[l]!=bk[x.l])return bk[l]<bk[x.l];
                if(bk[r]!=bk[x.r]){
                        return bk[r]<bk[x.r];
                        if(bk[l]&1){
                                return bk[r]>bk[x.r];
                        }else return bk[r]<bk[x.r];
                }
                if(bk[r]&1)return tim>x.tim;
                return tim<x.tim;
        }
}q[N];
struct C{
        int pos,val;
}c[N];
ll ans[N];
int bksiz;
int n;
ll num;
int a[N],number[N<<1];
int cnt[N<<1];
int h[N];

int L,R,Tim;
```

```
void add(int pos){
        int x=a[pos];
        h[cnt[x]]--;
        cnt[x]++;
        h[cnt[x]]++;
}

void del(int pos){
        int x=a[pos];
        h[cnt[x]]--;
        cnt[x]--;
        h[cnt[x]]++;
}

void cul(int tim){
        int pos=c[tim].pos;
        if(L<=pos&&pos<=R){

                del(pos);
                swap(a[pos],c[tim].val);
                add(pos);
                return;

        }
        swap(a[pos],c[tim].val);
}

int qnum,cnum,numnum;

void solv(){
        n=read();       m=read();
        for(int i=1;i<=n;i++)a[i]=number[i]=read();
        numnum=n;
        for(int i=1;i<=m;i++){
                int op=read(),l=read(),r=read();
                if(op==1){
                        qnum++;
                        q[qnum].id=qnum;
                        q[qnum].tim=cnum;
                        q[qnum].l=l;q[qnum].r=r;
                }else{
                        cnum++;
                        c[cnum].pos=l;
                        c[cnum].val=r;
                        number[++numnum]=r;
                }
        }

        sort(number+1,number+1+numnum);
```

```
        numnum=unique(number+1,number+1+numnum)-number-1;
        for(int i=1;i<=n;i++)a[i]=lower_bound(number+1,number+1+numnum,a[i])-number;
        for(int i=1;i<=cnum;i++)c[i].val=lower_bound(number+1,number+1+numnum,c[i].val)-number;

        bksiz=pow(n,0.666);
        for(int i=1;i<=n;i++)bk[i]=i/bksiz;
        sort(q+1,q+1+qnum);
        L=1,R=0,Tim=0;
        for(int i=1;i<=qnum;i++){
                int l=q[i].l,r=q[i].r,tim=q[i].tim;
                while(l<L)add(--L);
                while(r>R)add(++R);

                while(l>L)del(L++);
                while(r<R)del(R--);
                while(tim>Tim)cul(++Tim);
                while(tim<Tim)cul(Tim--);

                for(int j=1;j<=n;j++){
                        if(h[j]==0){
                                ans[q[i].id]=j;
                                break;
                        }
                }
        }
        for(int i=1;i<=qnum;i++){
                printf("%lld\n",ans[i]);
        }
}


int main(){
        int tst=1;
        while(tst--){
                solv();
        }
    return 0;
}
```

树上莫队

求a到b的简单路径上，有多少种不同的颜色
空间记得开2倍

```
#pragma GCC optimize(2)
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=1e6+9;
ll read(){
	ll ans=0,f=1;
	char c=getchar();
	while(c<'0'||c>'9'){
		if(c=='-')f=-1;
		c=getchar();
	}
	while(c>='0'&&c<='9'){
		ans=(ans<<1)+(ans<<3)+c-'0';
		c=getchar();
	}
	return ans*f;
}
struct line{
	int to,nex;
}l[N<<1];
int fir[N],cntline;
void addline(int fr,int to){
	cntline++;
	l[cntline].to=to;
	l[cntline].nex=fir[fr];
	fir[fr]=cntline;
}
int wson[N],f[N],siz[N],dep[N],ord[N<<1],dfn[N][2],dfncnt;

void dfssiz(int now,int fa){
	dfn[now][0]=++dfncnt;
	ord[dfncnt]=now;
	f[now]=fa;
	dep[now]=dep[fa]+1;
	siz[now]=1;
	wson[now]=0;
	for(int i=fir[now];i;i=l[i].nex){
		int to=l[i].to;
		if(to==fa)continue;
		dfssiz(to,now);
		siz[now]+=siz[to];
		if(siz[wson[now]]<siz[to])wson[now]=to;
	}
	dfn[now][1]=++dfncnt;
	ord[dfncnt]=now;
}

int top[N];
```

```
void dfstop(int now,int op){
        top[now]=op;
        if(wson[now])dfstop(wson[now],op);
        for(int i=fir[now];i;i=l[i].nex){
                int to=l[i].to;
                if(to==f[now]||to==wson[now])continue;
                dfstop(to,to);
        }
}
int lca(int a,int b){
        while(top[a]!=top[b]){
                if(dep[top[a]]<dep[top[b]])swap(a,b);
                a=f[top[a]];
        }
        return dep[a]<dep[b]?a:b;
}

int n,m;
int bk[N];
struct Q{
        int l,r;
        int lca;
        int id;
        bool operator<(Q x)const{
                if(bk[l]!=bk[x.l])return bk[l]<bk[x.l];
                if(bk[l]&1)return r<x.r;
                return r>x.r;
        }
}q[N];
ll num;
int a[N];
int w[N];
int qnum;
int bksiz;
int cnt[N];
ll ans[N];
int used[N];
void add(int pos){
        if(++cnt[w[pos]]==1)num++;
}
void del(int pos){
        if(--cnt[w[pos]]==0)num--;
}
void calc(int x){
        if(used[x])del(x);
        else add(x);
        used[x]^=1;
}
```

```
int main(){
        n=read(),m=read();
        for(int i=1;i<=n;i++)w[i]=a[i]=read();
        sort(a+1,a+1+n);
        int anum=unique(a+1,a+1+n)-a-1;
        for(int i=1;i<=n;i++)w[i]=lower_bound(a+1,a+1+anum,w[i])-a-1;
        for(int i=1;i<n;i++){
                int fr=read(),to=read();
                addline(fr,to);
                addline(to,fr);
        }
        dfssiz(1,1);
        dfstop(1,0);
        for(int i=1;i<=m;i++){
                        qnum++;
                        int a=read(),b=read();
                        if(dfn[a][0]>dfn[b][0])swap(a,b);
                        q[qnum].lca=lca(a,b);
                        q[qnum].id=qnum;
                        if(q[qnum].lca==a){
                                q[i].l=dfn[a][0];
                                q[i].r=dfn[b][0];
                                q[i].lca=0;
                        }else{
                                q[i].l=dfn[a][1];
                                q[i].r=dfn[b][0];
                        }
        }
        bksiz=sqrt(n);
        n<<=1;
        for(int i=1;i<=n;i++)bk[i]=i/bksiz;
        n>>=1;
        sort(q+1,q+1+qnum);
        int L=1,R=0;
        for(int i=1;i<=qnum;i++){
                int ql=q[i].l,qr=q[i].r,LCA=q[i].lca;

                while(ql<L)calc(ord[--L]);
                while(ql>L)calc(ord[L++]);
                while(qr>R)calc(ord[++R]);
                while(qr<R)calc(ord[R--]);
                if(LCA)calc(LCA);
                ans[q[i].id]=num;
                if(LCA)calc(LCA);
        }
        for(int i=1;i<=qnum;i++){
                printf("%lld\n",ans[i]);
        }
```

```
    return 0;
}
```

回滚莫队

区间[l,r],相同数字之间距离的最大值

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N=2e5+9;
const int mod=1e9+7;
ll read(){
        ll ans=0,f=1;
        char c=getchar();
        while(c<'0'||c>'9'){
                if(c=='-')f=-1;
                c=getchar();
        }
        while(c>='0'&&c<='9'){
                ans=(ans<<1)+(ans<<3)+c-'0';
                c=getchar();
        }
        return ans*f;
}
int n,m;
int bksiz,bknum;
int bk[N];
int lbk[N],rbk[N];
struct Q{
        int l,r;
        int id;
        bool operator<(const Q x)const{
                if(bk[l]!=bk[x.l])return bk[l]<bk[x.l];
                return r<x.r;
        }
}q[N];
ll a[N];
ll ans[N];
ll num;
int ml1[N],ml2[N];
int mr1[N],mr2[N];
ll number[N];
int qnum,numnum;

void solv(){
        n=read();
        for(int i=1;i<=n;i++)a[i]=number[i]=read();
        numnum=n;
        m=read();
        for(int i=1;i<=m;i++){
                int l=read(),r=read();

                        qnum++;
                        q[qnum].id=qnum;
                        q[qnum].l=l;q[qnum].r=r;
```

```
	}

	sort(number+1,number+1+numnum);
	numnum=unique(number+1,number+1+numnum)-number-1;
	for(int i=1;i<=n;i++)a[i]=lower_bound(number+1,number+1+numnum,a[i])-number;

	bksiz=sqrt(n);
	bknum=n/bksiz;
	if(n%bksiz)bknum++;
	for(int i=1;i<=bknum;i++){
		lbk[i]=rbk[i-1]+1;
		rbk[i]=bksiz*i;
	}
	rbk[bknum]=n;
	for(int i=1;i<=n;i++)bk[i]=(i-1)/bksiz+1;

	sort(q+1,q+1+qnum);

	int now=1;
	for(int i=1;i<=bknum;i++){
		int L=rbk[i]+1,R=rbk[i];
		int rans=0;
		for(int j=0;j<=numnum;j++)ml2[j]=ml1[j]=1e9,mr2[j]=mr1[j]=-1e9;

		for(now;bk[q[now].l]==i;now++){
			int l=q[now].l,r=q[now].r;
			int lans=0;
			if(bk[l]==bk[r]){
				for(int j=l;j<=r;j++){
					int id=a[j];
					ml1[id]=min(ml1[id],j);
					mr1[id]=max(mr1[id],j);
					lans=max(lans,mr1[id]-ml1[id]);
				}
				for(int j=l;j<=r;j++)ml1[a[j]]=1e9,mr1[a[j]]=-1e9;
				ans[q[now].id]=lans;
				continue;
			}
			while(R<r){
				int id=a[++R];
				ml2[id]=min(ml2[id],R);
				mr2[id]=max(mr2[id],R);
				rans=max(rans,mr2[id]-ml2[id]);
			}
			lans=rans;
			while(L>l){
				int id=a[--L];
				ml1[id]=min(ml1[id],L);
```

```
                                mr1[id]=max(mr1[id],L);
                                lans=max(lans,max(mr1[id],mr2[id])-ml1[id]);
                        }
                        while(L<rbk[i]+1){
                                int id=a[L++];
                                ml1[id]=1e9,mr1[id]=-1e9;
                        }
                        ans[q[now].id]=lans;
                }
        }
        for(int i=1;i<=qnum;i++){
                printf("%lld\n",ans[i]);
        }
}


int main(){
        int tst=1;
        while(tst--){
                solv();
        }
    return 0;
}
```

树状数组BitTree

单点修改，区间查询

```
int c[N];
//单点修改
void add(int pos,int k){
    for (int i = pos;i <= n;i += i&-i) c[i] += k;
}
//区间查询
int query(int x){
    int ans = 0;
    for (int i = x;i;i -= i&-i) ans += c[i];
    return ans;
}
int query(int l,int r){
    return query(r)-query(l-1);
}
```

区间修改，单点查询(差分)

```
int c[N];
//区间修改
void add(int pos,int k){
    for (int i = pos;i <= n;i += i&-i) c[i] += k;
}

void add(int l,int r,int x){
    add(l,x);
    add(r+1,-x);
}
//单点查询
int query(int x){
    int ans = 0;
    for (int i = x;i;i -= i&-i) ans += c[i];
    return ans;
}
```

区间修改，区间查询(没验证)

```
int c1[N],c2[N];
//区间修改
void add(int pos,int k)
{
    for (int i = pos;i <= n;i += i&-i) c1[i] += k,c2[i] += pos * k;
}
void add(int l,int r,int k){
    add(l,k);
    add(r+1,-k);
}
//区间查询
int query(int pos)
{
    int ans = 0;
    for (int i = pos;i;i -= i&-i)
    {
        ans += (pos + 1) * c1[i]- c2[i];
    }
    return ans;
}
int query(int l,int r){
    return query(r)-query(l-1);
}
```

并查集相关

可删除并查集

```
int fa[N];
int n,fanum;
int find(int x){
        if(fa[x]==x)return x;
        return fa[x]=find(fa[x]);
}
void merge(int x,int y){
        x=find(x),y=find(y);
        fa[x]=y;
}
void del(int x){
        fa[++fanum]=fanum;
        fa[x]=fanum;
}
void init(){
        fanum=n;
        for(int i=1;i<=n;i++){
                del(i);
        }
}
```

带权并查集

食物链

- `1 X Y`，表示 X 和 Y 是同类。
- `2 X Y`，表示 X 吃 Y。

输出假话的总数。

```
int fa[N],d[N];
int n,k;
int find(int x){
        if(x==fa[x])return x;
        int F=fa[x];
        fa[x]=find(fa[x]);
        d[x]+=d[F];
        return fa[x];
}

void merge(int x,int y,int v){
        int fx=find(x),fy=find(y);
        if(fx==fy)return;
        fa[fx]=fy;
        d[fx]=d[y]-d[x]+v;
}

void solv(){
        cin>>n>>k;
        for(int i=1;i<=n;i++){
                fa[i]=i;d[i]=0;
        }
        int ans=0;
        for(int i=1;i<=k;i++){
                int a,b,ax;
                cin>>ax>>a>>b;
                if(a>n||b>n||a==b&&ax==2){
                        ans++;
                        continue;
                }
                int A=find(a),B=find(b);
                if(A==B){
                        int gx=(d[a]-d[b])%3+3;
                        if(ax==1&&gx%3||ax==2&&gx%3!=1){
                                ans++;
                        }
                        continue;
                }
                if(ax==1)merge(a,b,0);
                else merge(a,b,1);
        }
        cout<<ans<<'\n';
}
```

可持久化并查集

```
struct Point{
        int lson,rson;
        int f,dep;
}p[N*40];
int rt[N<<1];
int pnum;
void build(int &now,int l,int r){
        now=++pnum;
        if(l==r){
                p[now].f=l;
                return;
        }
        int mid=l+r>>1;
        build(p[now].lson,l,mid);
        build(p[now].rson,mid+1,r);
}
void merge(int &now,int pre,int l,int r,int pos,int k,int dd){
        now=++pnum;
        p[now]=p[pre];
        if(l==r){
                p[now].f=k;
                p[now].dep+=dd;
                return;
        }
        int mid=l+r>>1;
        if(pos<=mid){
                merge(p[now].lson,p[pre].lson,l,mid,pos,k,dd);
        }else{
                merge(p[now].rson,p[pre].rson,mid+1,r,pos,k,dd);
        }
}
int query(int now,int l,int r,int pos){
        if(l==r){
                return now;
        }
        int mid=l+r>>1;
        if(pos<=mid)return query(p[now].lson,l,mid,pos);
        else return query(p[now].rson,mid+1,r,pos);
}
int find(int rt,int x){
        int pos=query(rt,1,n,x);
        if(p[pos].f==x)return pos;
        return find(rt,p[pos].f);
}

void solv(){
```

```
        cin>>n>>m;
        build(rt[0],1,n);
        for(int i=1;i<=m;i++){
                int opt,x,y;
                cin>>opt>>x;
                if(opt==1){
                        cin>>y;
                        int posx=find(rt[i-1],x),posy=find(rt[i-1],y);
                        rt[i]=rt[i-1];
                        if(p[posx].f!=p[posy].f){
                                if(p[posx].dep>p[posy].dep)swap(posx,posy);
                                merge(rt[i],rt[i-1],1,n,p[posx].f,p[posy].f,0);
                                if(p[posx].dep==p[posy].dep){
                                        int u=rt[i];
                                        merge(rt[i],u,1,n,p[posy].f,p[posy].f,1);
                                }
                        }
                }else if(opt==2){
                        rt[i]=rt[x];
                }else if(opt==3){
                        cin>>y;
                        int posx=find(rt[i-1],x),posy=find(rt[i-1],y);
                        if(p[posx].f==p[posy].f){
                                cout<<"1\n";
                        }else cout<<"0\n";
                        rt[i]=rt[i-1];
                }else{
                        cout<<"WA\n";
                        while(1);
                }
        }
}
```

并查集+2Sat思想

给定长n的01串s, 给定k个集合A1,...,Ak,保证任意三个集合交集为空. 每次操作选择一个集合,翻转s中对应位置. 定义mi为使前i个位置全为1所需的最少操作数(题目数据保证每个mi都存在), 求所有mi的值.

```
int n,m;

int fa[N],siz[N];
int find(int x){
        if(x==fa[x])return x;
        return fa[x]=find(fa[x]);
}
void unin(int a,int b){
        a=find(a),b=find(b);
        if(a==b)return;
        fa[a]=b;siz[b]+=siz[a];
}
int sel(int x){
        return min(siz[find(x)],siz[find(x+m)]);
}

char c[N];
vector<int>vec[N];
void solv(){
        n=read(),m=read();
        for(int i=1;i<=m*2;i++){
                fa[i]=i;
                if(i<=m)siz[i]=1;
                else siz[i]=0;
        }
        siz[m*2+1]=1e9;
        fa[m*2+1]=m*2+1;
        for(int i=1;i<=n;i++){
                vec[i].clear();
        }
        scanf("%s",c+1);
        for(int i=1;i<=m;i++){
                int r=read();
                for(int j=1;j<=r;j++){
                        int id=read();
                        vec[id].push_back(i);
                }
        }
        int nowans=0;
        for(int i=1;i<=n;i++){
                if(vec[i].size()==1){
                        int ax=vec[i][0];
                        nowans-=sel(ax);
                        if(c[i]=='0'){
                                unin(ax+m,m*2+1);
                        }else{
                                unin(ax,m*2+1);
                        }
                        nowans+=sel(ax);
```

```
                }else if(vec[i].size()==2){
                        int a=vec[i][0],b=vec[i][1];
                        if(find(a)==find(b)||find(a+m)==find(b)){

                        }else{
                                nowans-=sel(a)+sel(b);
                                if(c[i]=='0'){
                                        unin(a+m,b);unin(a,b+m);
                                }else{
                                        unin(a,b);unin(a+m,b+m);
                                }
                                nowans+=sel(a);
                        }
                }
                printf("%d\n",nowans);
        }
}
```

主席树

求区间mex

```
#include<bits/stdc++.h>
using namespace std;
const int ma=2e6+9;
int n,m,a[ma],t[ma],b[ma],n1;
struct point{
        int l,r,cnt;
}p[ma*50];
vector<int>v;
int rot[ma],cnt;
void insert(int &now,int pre,int l,int r,int k,int i){
        p[++cnt]=p[pre];
        now=cnt;
        if(l==r){
                p[now].cnt=i;
                return;
        }
        int mid=l+r>>1;
        if(mid>=k){
                insert(p[now].l,p[pre].l,l,mid,k,i);
        }else insert(p[now].r,p[pre].r,mid+1,r,k,i);
        p[now].cnt=min(p[p[now].l].cnt,p[p[now].r].cnt);
}
int query(int now,int dl,int l,int r){
        if(now==0||l==r){
                return l;
        }
        int mid=l+r>>1;
        if(p[p[now].l].cnt<dl){
                return query(p[now].l,dl,l,mid);
        }else return query(p[now].r,dl,mid+1,r);
}
int main(){
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;i++){
        scanf("%d",&a[i]);
        t[i]=a[i];
        t[n+i]=a[i]+1;
        }
        sort(t+1,t+2*n+2);
        n1=unique(t+1,t+2*n+2)-t-1;
        for(int i=1;i<=n;i++){
                b[i]=lower_bound(t+1,t+n1+1,a[i])-t;
```

```
                insert(rot[i],rot[i-1],1,n1,b[i],i);
        }
        for(int i=1;i<=m;i++){
                int l,r,tp;
        scanf("%d %d",&l,&r);
        tp=query(rot[r],l,1,n1);
        cout<<t[tp]<<endl;
        }
        return 0;
}
```

[SDOI2013] 森林

题目描述

小 Z 有一片森林，含有 N 个节点，每个节点上都有一个非负整数作为权值。初始的时候，森林中有 M 条边。

小Z希望执行 T 个操作，操作有两类：

- `Q x y k` 查询点 x 到点 y 路径上所有的权值中，第 k 小的权值是多少。此操作保证点 x 和点 y 连通，同时这两个节点的路径上至少有 k 个点。
- `L x y` 在点 x 和点 y 之间连接一条边。保证完成此操作后，仍然是一片森林。

为了体现程序的在线性，我们把输入数据进行了加密。设 $lastans$ 为程序上一次输出的结果，初始的时候 $lastans$ 为 0。

样例输入 #1

```
1
8 4 8
1 1 2 2 3 3 4 4
4 7
1 8
2 4
2 1
Q 8 7 3
Q 3 5 1
Q 10 0 0
L 5 4
L 3 2
L 0 7
Q 9 2 5
Q 6 1 6
```

样例输出 #1

```
2
2
1
4
2
```

```
const int N=1e5+9;
struct Point{
        int lson,rson;
        int cnt;
}p[N*100];
int pnum;
vector<int>l[N];
int n,m,q;
int number[N],numbernum;
int w[N];

void change(int &now,int pre,int l,int r,int pos,int k){
        now=++pnum;
        p[now]=p[pre];
        p[now].cnt+=k;
        if(l==r){
                return;
        }
        int mid=l+r>>1;
        if(pos<=mid)change(p[now].lson,p[pre].lson,l,mid,pos,k);
        else change(p[now].rson,p[pre].rson,mid+1,r,pos,k);
}

int query(int a,int b,int c,int d,int l,int r,int num){
        if(l==r){
                return l;
        }
        int mid=l+r>>1;
        int lnum=p[p[a].lson].cnt+p[p[b].lson].cnt-p[p[c].lson].cnt-p[p[d].lson].cnt;
        if(num<=lnum){
                return query(p[a].lson,p[b].lson,p[c].lson,p[d].lson,l,mid,num);
        }else return query(p[a].rson,p[b].rson,p[c].rson,p[d].rson,mid+1,r,num-lnum);
}
int siz[N],dep[N];
int rt[N],bkrt[N];
int f[N][22];
void dfs(int now,int fa,int tp){
        bkrt[now]=tp;siz[tp]++;
        change(rt[now],rt[fa],1,numbernum,w[now],1);
        f[now][0]=fa;dep[now]=dep[fa]+1;
        for(int i=1;i<=19;i++){
                f[now][i]=f[f[now][i-1]][i-1];
        }
        for(auto to:l[now]){
                if(to==fa)continue;
                dfs(to,now,tp);
        }
}
```

```
int LCA(int a,int b){
        if(dep[a]<dep[b])swap(a,b);
        for(int i=19;i>=0;i--){
                if(dep[f[a][i]]>=dep[b])a=f[a][i];
        }
        if(a==b)return b;
        for(int i=19;i>=0;i--){
                if(f[a][i]!=f[b][i]){
                        a=f[a][i],b=f[b][i];
                }
        }
        return f[a][0];
}

void solv(){
        cin>>n>>m>>q;
        for(int i=0;i<=n;i++){
                l[i].clear();
                siz[i]=0;
                bkrt[i]=i,rt[i]=0;
        }
        for(int i=1;i<=n;i++){
                cin>>w[i];
                number[i]=w[i];
        }
        numbernum=n;
        sort(number+1,number+1+numbernum);
        numbernum=unique(number+1,number+1+numbernum)-number-1;
        for(int i=1;i<=n;i++){
                w[i]=lower_bound(number+1,number+1+numbernum,w[i])-number;
        }
        pnum=0;

        for(int i=1;i<=m;i++){
                int a,b;
                cin>>a>>b;
                l[a].push_back(b);
                l[b].push_back(a);
        }
        for(int i=1;i<=n;i++){
                if(bkrt[i]==i)dfs(i,0,i);
        }


        int lastans=0;
        for(int i=1;i<=q;i++){
                char opt;
                cin>>opt;
                int x,y;
```

```
                cin>>x>>y;
                x^=lastans,y^=lastans;
                if(opt=='Q'){
                        int k;
                        cin>>k;
                        k^=lastans;
                        int anc=LCA(x,y),faanc=f[anc][0];
                        lastans=query(rt[x],rt[y],rt[anc],rt[faanc],1,numbernum,k);
                        lastans=number[lastans];
                        cout<<lastans<<'\n';
                }else if(opt=='L'){
                        l[x].push_back(y);
                        l[y].push_back(x);
                        if(siz[bkrt[x]]>siz[bkrt[y]]){
                                dfs(y,x,bkrt[x]);
                        }else{
                                dfs(x,y,bkrt[y]);
                        }
                }else{
                        cout<<"WA"<<endl;
                        while(1);
                }
        }
}
```

区间数颜色（在线）

[SDOI2009] HH的项链

输入格式

一行一个正整数 n，表示项链长度。
第二行 n 个正整数 a_i，表示项链中第 i 个贝壳的种类。

第三行一个整数 m，表示 HH 询问的个数。
接下来 m 行，每行两个整数 l, r，表示询问的区间。

```
const int N=1e6+9;
const int mod=1e9+7;
struct Point{
        int lson,rson;
        int sum;
}p[N*40];
int pnum;
void pushup(int now){
        p[now].sum=p[p[now].lson].sum+p[p[now].rson].sum;
}
void change(int &now,int pre,int l,int r,int pos,int val){
        now=++pnum;
        p[now]=p[pre];
        if(l==r){
                p[now].sum+=val;
                return;
        }
        int mid=l+r>>1;
        if(pos<=mid){
                change(p[now].lson,p[pre].lson,l,mid,pos,val);
        }else{
                change(p[now].rson,p[pre].rson,mid+1,r,pos,val);
        }
        pushup(now);
}
int query(int now,int pos,int l,int r){
        if(pos<=l)return p[now].sum;
        int mid=l+r>>1;
        if(pos<=mid){
                return query(p[now].lson,pos,l,mid)+p[p[now].rson].sum;
        }else{
                return query(p[now].rson,pos,mid+1,r);
        }
}
int n;
int a[N];
int m;
int last[N];
int rot[N];
void solv(){
        cin>>n;
        int l=1,r=N-1;
        for(int i=1;i<=n;i++)cin>>a[i];
        for(int i=1;i<=n;i++){
                if(last[a[i]]){
                        int tmp=0;
                        change(tmp,rot[i-1],l,r,last[a[i]],-1);
                        last[a[i]]=i;
                        change(rot[i],tmp,l,r,last[a[i]],1);
```

```
                }else{
                        last[a[i]]=i;
                        change(rot[i],rot[i-1],l,r,last[a[i]],1);
                }
        }
        cin>>m;
        for(int i=1;i<=m;i++){
                int L,R;
                cin>>L>>R;
                cout<<query(rot[R],L,l,r)<<'\n';
        }
}
```

RMQ ST

```
int n,a[N];
int st[N][22],lg[N];
void init(){
        lg[0]=-1;
        for(int i=1;i<=n;i++){
                lg[i]=lg[i>>1]+1;
                st[i][0]=a[i];
        }
        for(int i=1;i<22;i++){
                int len=1<<i;
                for(int j=1;j+len-1<=n;j++){
                        st[j][i]=max(st[j][i-1],st[j+(len>>1)][i-1]);
                }
        }
}
int query(int l,int r){
        int k=lg[r-l+1];
        return max(st[l][k],st[r-(1<<k)+1][k]);
}
```